Encrypted Payloads in SUIT Manifests

draft-ietf-suit-firmware-encryption-21

Hannes Tschofenig, Russ Housley, Brendan Moran,
David Brown and Ken Takayama

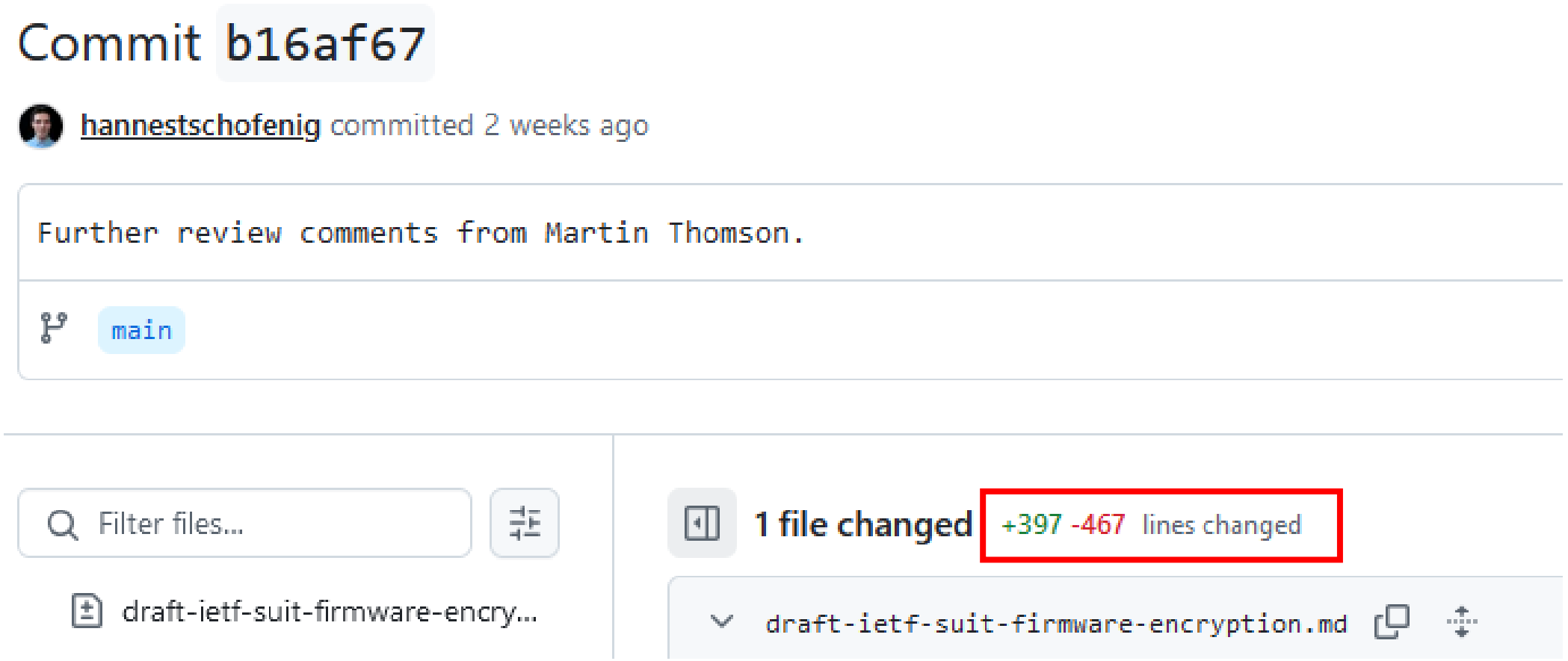
Status

- Reviewed
  - Almost ready by Martin Thomson
  - Ready by Ron Bonica
- Addressed comments
  - Most are editorial
- Now on IESG Evaluation

Review comment from Martin Thomson

(Editorial)

- Brush up

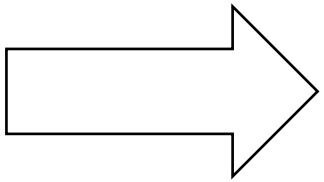


Review comment from Martin Thomson

(Editorial)

Section 6 introduces notation that looks like function invocations, except that it is not.
In LaTeX, I would write CEK(R, S) with a subscript as $¥mathsf{CEK}_{R,S}$.
This is somewhat confusing as ENC(K, V) is a function.
I recommend the use of square brackets.

1. Fetch KEK(*, S)
2. Generate CEK
3. ENC(CEK, KEK)
4. ENC(payload, CEK)



1. Fetch KEK[*, S]
2. Generate CEK
3. ENC(CEK, KEK)
4. ENC(payload, CEK)

Fixed References

(Very Editorial)

Before

In .md file

Section 4.3.17 of {{RFC9124}}.

Then in .html file

```
Section 4.3.17 of [RFC9124].
```

After

In .md file

{{Section 4.3.17 of RFC9124}}.

Then in .html file

```
Section 4.3.17 of [RFC9124].
```

Cool!

KDF Contexts for ECDH-ES+A128KW are Updated

(A bit affects mainly SUIT Manifest examples)

Before

- Algorithm ID: -65534 (A128CTR)
- SuppPubInfo
  - keyDataLength: 128
  - protected: {
    / alg / 1: -29 / ECDH-ES+A128KW /
    }
  - other: 'SUIT Payload Encryption'

After

- Algorithm ID: -3 (A128KW)
- SuppPubInfo
  - keyDataLength: 128
  - protected: {
    / alg / 1: -29 / ECDH-ES+A128KW /
    }
  - other: 'SUIT Payload Encryption'

RFC 9053 says only "either a key wrap algorithm identifier or a content encryption algorithm identifier" so both are allowed.
But we chose key wrap algorithm identifier since all ECDH+AES-KW examples in github.com/cose-wg/Examples uses key wrap algorithms. It must be widely supported and tested.

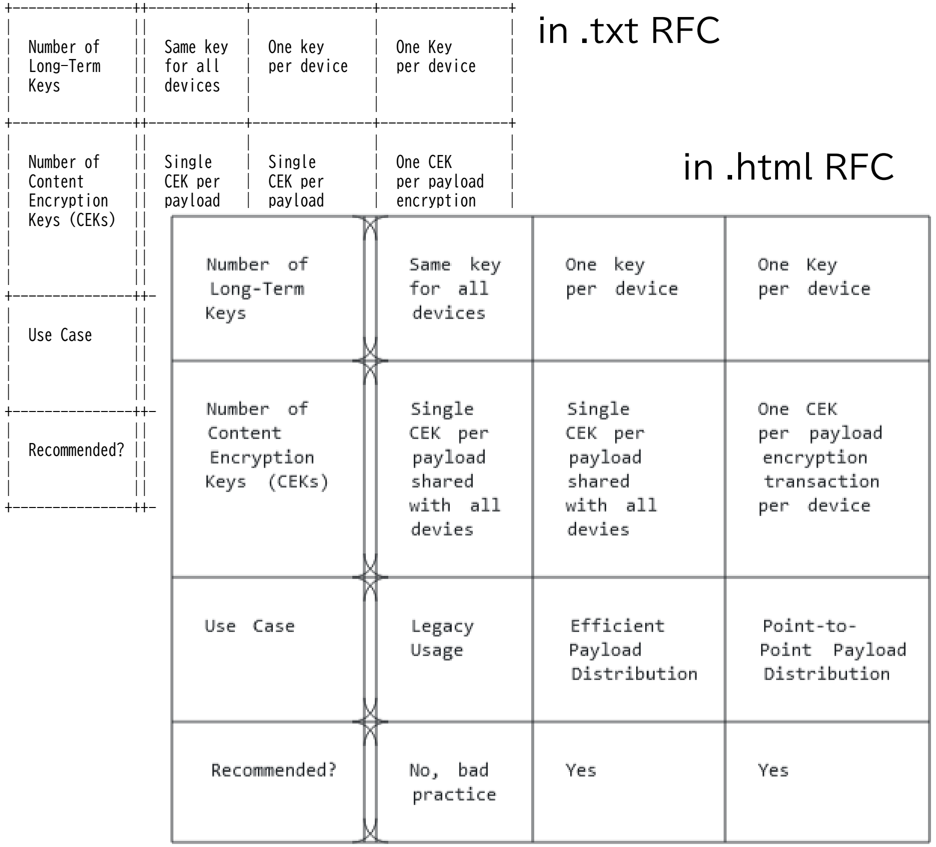
Addressed Comment on Table 3 by Martin Thomson

(Editorial, need to brush up)

- Section 12 has a wonderful table that is drawn using ascii art, which is then converted to an SVG. Please use a table.

in .txt RFC

in .html RFC



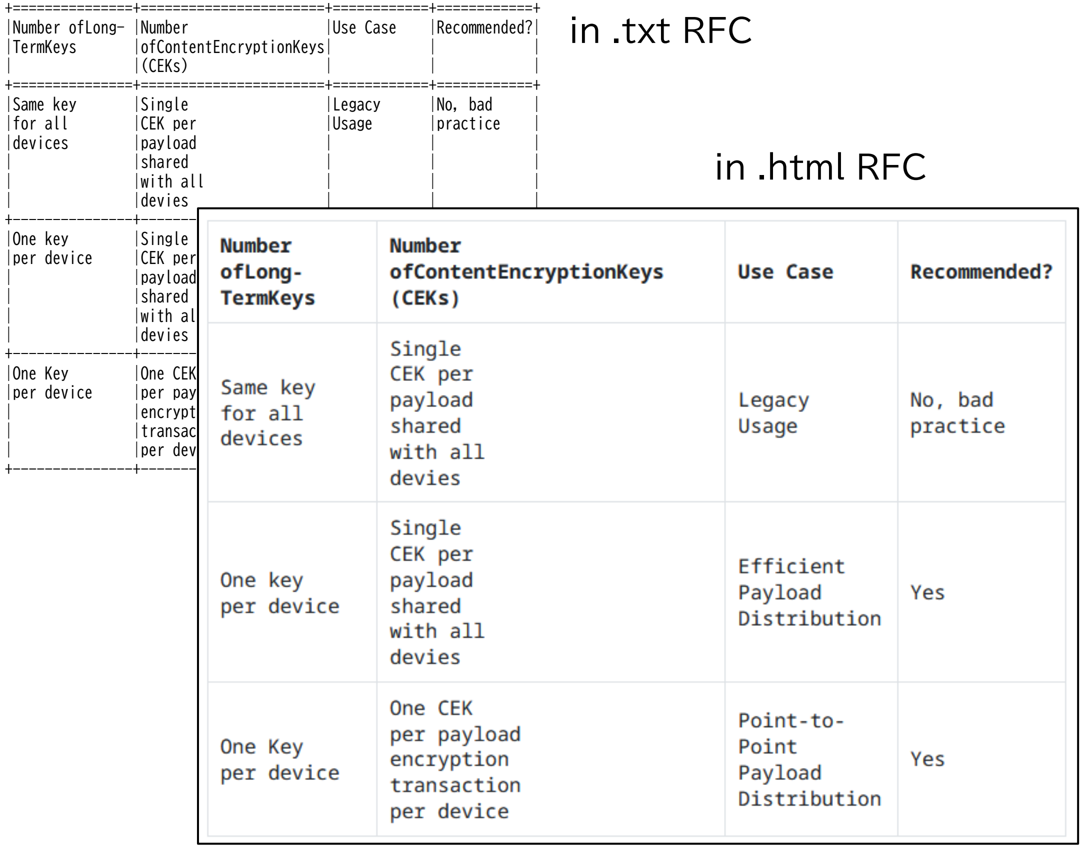
Number of Long-Term Keys	Same key for all devices	One key per device	One Key per device
Number of Content Encryption Keys (CEKs)	Single CEK per payload shared with all devies	Single CEK per payload shared with all devies	One CEK per payload encryption transaction per device
Use Case	Legacy Usage	Efficient Payload Distribution	Point-to-Point Payload Distribution
Recommended?	No, bad practice	Yes	Yes

Previously, not a table but a figure

Use a Table and Transform

in .txt RFC

in .html RFC



Number ofLong-TermKeys	Number ofContentEncryptionKeys (CEKs)	Use Case	Recommended?
Same key for all devices	Single CEK per payload shared with all devies	Legacy Usage	No, bad practice
One key per device	Single CEK per payload shared with all devies	Efficient Payload Distribution	Yes
One Key per device	One CEK per payload encryption transaction per device	Point-to-Point Payload Distribution	Yes

Now it is *Table 3*

Addressed Too Long Line Warning

(Editorial)

Before

```
recipient_header_unpr_map_aeskw =
{
    1 => int,       ; algorithm identifier
  ? 4 => bstr,      ; identifier of the KEK pre-shared with the recipient
  * label => values   ; extension point
}
```

72 characters are the limit for code block

✅After

```
recipient_header_unpr_map_aeskw =
{
    1 => int,       ; algorithm identifier
  ? 4 => bstr,      ; identifier of the KEK pre-shared with the recipient
  * label => values   ; extension point
}
```

SUIT Manifest examples are also aligned to the 72 characters.